Available online at www.sciencedirect.com

ScienceDirect

Mathematical and Computer Modelling 48 (2008) 1254–1264

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Improved MILP models for two-machine flowshop with batch processing machines

Ching-Jong Liao[a,*], Li-Man Liao[b]

[a] *Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan*
[b] *Department of Industrial Engineering and Management, National Chin-Yi University of Technology, Taichung, Taiwan*

Received 13 March 2007; received in revised form 27 December 2007; accepted 11 January 2008

Abstract

In this paper, we consider the problem of scheduling jobs in a flowshop with two batch processing machines such that the makespan is minimized. Batch processing machines are frequently encountered in many industrial environments such as heat treatment operations in a steel foundry and chemical processes performed in tanks or kilns. Improved Mixed Integer Linear Programming (MILP) models are presented for the flowshop problem with unlimited or zero intermediate storage. An MILP-based heuristic is also developed for the problem. Computational experiments show that the new MILP models can significantly improve the original ones. Also, the heuristic can obtain the optimal solutions for all the test problem instances.

Keywords: Batch processing; Scheduling; Flow shop; Mathematical formulation

1. Introduction

In this paper, we consider the problem of scheduling batch processing machines in a two-machine flowshop. A batch processing machine is one that can process several jobs simultaneously as long as the total size of the batch of jobs does not exceed the machine capacity. Batch processing machines are frequently encountered in many industrial environments such as heat treatment operations in a steel foundry and chemical processes performed in tanks or kilns [4]. Other applications of batch processing machines can be found in Uzsoy [8] and Damodaran and Srihari [2]. Uzsoy [8] describes an application for burn-in operations in semiconductor manufacturing. Damodaran and Srihari [2] provides an application of batch processing machines used in chambers for environmental stress screening in a printed circuit board assembly environment.

The addressed problem can be defined as follows. There is a set of n jobs ($j \in J$) to be grouped into batches ($b \in B$). The batches of jobs are then to be processed on two machines ($m \in M$) in a flowshop. Each job j

* Corresponding address: Department of Industrial Management, National Taiwan University of Science and Technology, 43 Keelung Road, Section 4, Taipei 106, Taiwan. Fax: +886 2 27376344.

E-mail address: cjl@im.ntust.edu.tw (C.-J. Liao).

0895-7177/$ - see front matter
doi:10.1016/j.mcm.2008.01.001

has a processing time p_{jm} and a size s_j on machine m. All jobs in a batch begin processing at the same time, and the processing time of a batch P_{bm} is determined by the longest processing time of all the jobs in the batch, i.e., $P_{bm} = \max_{j \in b}\{p_{jm}\}$. Each machine m can process a batch of jobs simultaneously as long as the total size of the batch does not exceed the machine capacity S_m. The criterion to be minimized is the makespan, or the completion time of the last batch on machine 2.

The assumptions made for the problem are summarized as follows. All jobs are available for processing at time zero, and both machines are continuously available during the planning horizon. At any time each batch of jobs can be processed by at most one machine. The processing times of jobs are known and fixed, which include the required sequence-independent setup times. Preemption of batches or jobs is not allowed, i.e., once a batch is started on a machine it will continue in processing until the whole batch is completed. Two models are considered in this paper. The first model assumes that jobs are allowed to wait between successive machines and the intermediate storage is unlimited. The second model assumes that there are no intermediate storages.

The addressed problem is a permutation flowshop problem because there are optimal schedules for minimizing makespan in a two-machine flowshop that do not require sequence changes between machines [5]. Nevertheless, the problem is still NP-hard because the single machine version of this problem is NP-hard [8].

In the remaining of this section, we briefly review the related research on batch processing machines (or simply batching machines). There is an extensive literature on research that involves both batching and scheduling decisions. This line of research can be classified into two major models: family scheduling model and batching machine model [6]. In a family scheduling model, jobs are grouped into families so that no setup is required for a job if it belongs to the same family of the previously processed job. In a batching machine model, the batching machine is capable of processing several jobs simultaneously. For the single batching machine, Uzsoy [8] proposes a number of heuristics to minimize the makespan and the total completion time. Also, Brucker et al. [1] propose dynamic programming algorithms to optimize several different criteria both for unrestricted batch sizes, and for batches that can contain at most n jobs. The dynamic programming algorithms are further extended to identical parallel batching machines for unrestricted batch sizes. For the case of flow shop comprising of batching machines, Danneberg et al. [3] propose constructive algorithms to minimize the makespan under the assumption that the batches to be processed are given as an input. Sung and Kim [7] analyze a two-machine flow shop comprising of batching machines with respect to three due date related problems. The batching machines can process jobs simultaneously as long as the number of jobs in the batch is less than a predetermined number. Moreover, Damodaran and Srihari [2] propose mixed integer linear programming (MILP) models to minimize makespan in a two-machine flow shop with batching machines when the buffer capacities are unlimited or zero.

In this paper, we consider the same problem as Damodaran and Srihari [2]. We will propose improved MILP models for the problem. A heuristic procedure based on MILP will also be developed to derive near-optimal solutions in much less computation time.

Definition of the problem sets, parameters, and variables is as follows:

Sets

J	jobs
M	machines
B	batches
K	positions

Parameters

p_{jm}	processing time of job j on machine m
s_j	size of job j
S_m	capacity of machine m
$S_{\min}$	the smallest capacity of m machines, i.e., $S_{\min} = \min\{S_m \mid m = 1, 2\}$
$\#B$	number of batches

Decision variables

X_{jb}	1, if job j is in batch b; 0 otherwise
Z_{bk}	1, if batch b is scheduled in the kth position; 0 otherwise

Dependent variables

$C_{\max}$	makespan

P_{bm} processing time of batch b on machine m
C_{km} completion time of the kth batch on machine m
D_{km} departure time of the kth batch on machine m
Q_{km} processing time of the kth batch on machine m

2. Problem with unlimited intermediate storage

In this section, we consider the problem with unlimited intermediate storage.

2.1. DS1 model

Damodaran and Srihari [2] present the following MILP model (DS1 model) for the unlimited intermediate storage case:

$$\text{Minimize } C_{\max} \tag{1}$$

$$\text{subject to } \sum_{b \in B} X_{jb} = 1 \quad \forall j \in J \tag{2}$$

$$\sum_{j \in J} s_j X_{jb} \le S_{\min} \quad \forall b \in B \tag{3}$$

$$P_{bm} \ge p_{jm} X_{jb} \quad \forall j \in J, b \in B, m \in M \tag{4}$$

$$\sum_{k \in K} Z_{bk} = 1 \quad \forall b \in B \tag{5}$$

$$\sum_{b \in B} Z_{bk} = 1 \quad \forall k \in K \tag{6}$$

$$Q_{km} \ge P_{bm} + M(Z_{bk} - 1) \quad \forall b \in B, k \in K, m \in M \tag{7}$$

$$C_{k1} = \sum_{k'=1}^{k} Q_{k'1} \quad \forall k \in K \tag{8}$$

$$C_{12} = C_{11} + Q_{12} \tag{9}$$

$$C_{k2} \ge C_{k-1,2} + Q_{k2} \quad \forall k \in K/\{1\} \tag{10}$$

$$C_{k2} - C_{k1} \ge Q_{k2} \quad \forall k \in K \tag{11}$$

$$C_{\max} \ge C_{n2} \tag{12}$$

$$X_{jb}, Z_{bk} = 0 \text{ or } 1 \tag{13}$$

$$C_{km}, Q_{km}, P_{bm}, C_{\max} \ge 0. \tag{14}$$

Objective (1) minimizes the makespan. Constraint (2) ensures that each job is assigned to exactly one batch. Constraint (3) requires that the total size of jobs assigned to a batch cannot exceed the capacity of each machine. Constraint (4) is used to compute the processing time of batch b on machine m. Constraints (5) and (6) ensure that each batch is assigned to a position and each position has one batch assigned to it. Constraint (7) is used to compute the processing time of the kth batch on machine m. Constraints (8)–(11) are used to compute the completion times of the kth batch on the two machines. Constraint (12) determines the makespan. Constraints (13) and (14) impose the binary and nonnegativity restrictions on the variables.

2.2. Improved DS1 model

The above DS1 model introduces a variable Q_{km} in order to compute the completion time of each job. To define Q_{km}, another set of binary variables Z_{bk} is added. However, a careful examination of the DS1 model reveals that the variable Q_{km} is redundant. We can simply use the existing variable P_{bm} to compute the completion times of jobs. The MILP model will make a suitable arrangement so that P_{bm} is placed in the right position. Thus, constraints (5)–(7)

can be deleted. Accordingly, Q_{km} and C_{km} in constraints (8)–(11) need to be replaced by P_{bm} and C_{bm}, respectively. For completeness, the improved DS1 model is given as follows:

$$\text{Minimize } C_{\max} \tag{1'}$$

$$\text{Subject to } \sum_{b \in B} X_{jb} = 1 \quad \forall j \in J \tag{2'}$$

$$\sum_{j \in J} s_j X_{jb} \le S_{\min} \quad \forall b \in B \tag{3'}$$

$$P_{bm} \ge p_{jm} X_{jb} \quad \forall j \in J, b \in B, m \in M \tag{4'}$$

$$C_{b1} = \sum_{k=1}^{b} P_{k1} \quad \forall b \in B \tag{5'}$$

$$C_{12} = C_{11} + P_{12} \tag{6'}$$

$$C_{b2} \ge C_{b-1,2} + P_{b2} \quad \forall b \in B/\{1\} \tag{10'}$$

$$C_{b2} - C_{b1} \ge P_{b2} \quad \forall b \in B \tag{11'}$$

$$C_{\max} \ge C_{n2} \tag{12'}$$

$$X_{jb} = 0 \text{ or } 1 \tag{13'}$$

$$C_{bm}, P_{bm}, C_{\max} \ge 0. \tag{14'}$$

It is worth noting that the model with $\#B = n$ can be solved by the famous Johnson's algorithm [5]. Therefore, we can apply the MILP model with only $\#B = n - 1$ and compare with the solution from Johnson's algorithm for $\#B = n$ to obtain the optimal solution.

2.3. Lower bounds

Good lower bounds can help the MILP model be solved more efficiently. By considering each machine individually, we can establish a lower bound LB on the makespan as follows:

Step 1. For each machine m, determine the associated lower bound LB_m by performing Steps 2–4.

Step 2. Group the jobs, in LPT (longest processing time) order, into several batches by setting the total size of each batch as S_m (except the last batch), where jobs are allowed to be split.

Step 3. Sum up the processing times of the first jobs in each batch.

Step 4. Compute LB_m by adding the smallest processing time on the other machine to the sum in Step 3.

Step 5. The lower bound LB can be determined by combining the two LB_m, i.e.,

$$LB = \max\{LB_1, LB_2\}.$$

The lower bound can be easily justified. We first sequence jobs in non-increasing order of processing times (LPT order) and then group the jobs, which are allowed to be split, such that each batch has a total size S_m. Next, we sum up the processing times of the first jobs in each batch. It is clear that any interchanges of jobs in the sequence and/or grouping the jobs in any other ways cannot decrease the value of the sum. Finally, the smallest processing time on the other machine is added to account for the computation of makespan. Thus, the steps result in a lower bound on makespan of the problem.

Example 1. Consider the same numerical example as in Damodaran and Srihari [2]. The processing times and sizes of jobs are given in Table 1. The machine capacities are assumed to be 10. Applying the above steps yields the following result (see Fig. 1):

$$\begin{aligned} LB &= \max\{LB_1, LB_2\} \\ &= \max\{15 + 10 + 7 + 3 + 1, 14 + 10 + 5 + 4 + 2\} \\ &= \max\{36, 35\} = 36. \end{aligned}$$

The optimal solution of this problem is $C^*_{\max} = 45$, so the ratio of $LB/C^*_{\max}$ is 0.8.

Table 1
Data for Example 1

j	1	2	3	4	5	6	7	8	9	10
p_{j1}	10	2	6	15	7	9	3	10	10	6
p_{j2}	14	9	10	1	12	4	5	8	5	9
s_j	5	2	3	4	5	3	5	1	4	4

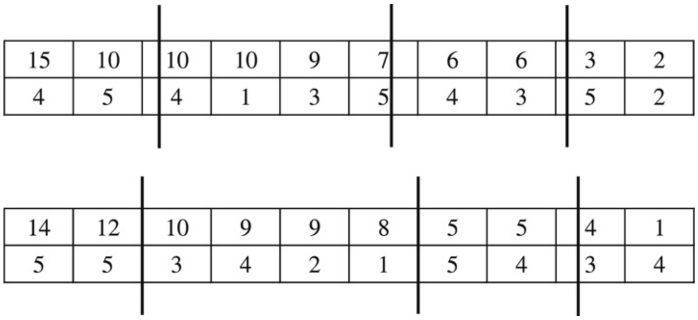

Fig. 1. Lower bound computation for Example 1.

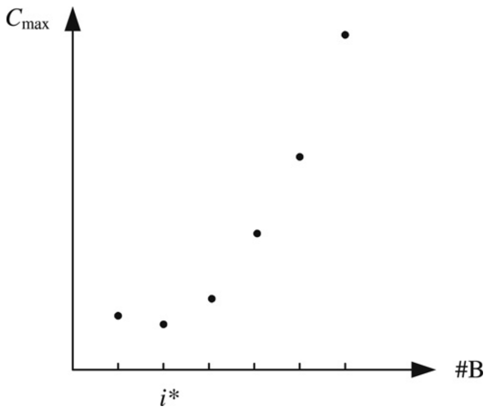

Fig. 2. The V-shape property.

2.4. Proposed heuristic

It is worth noting that in all the test problems the makespan has a V-shape property with respect to $\#B$, as depicted in Fig. 2. Let C^i_{max} be the makespan for the problem with a given $\#B = i$. Then the V-shape property can be stated as: if $C^{i+1}_{max} > C^i_{max}$, then $C^{i+2}_{max} > C^{i+1}_{max}$. The V-shape property can only be stated as a conjecture since we cannot prove it. But, on the other hand, we have not found any counter-example until now. Therefore, if only a near-optimal solution is desired, the solution procedure can be terminated once a local minimum makespan has been found. Based on the computational experiments given in Section 4, the V-shape property holds for all the test problems.

Let n_{LS} denote the number of large jobs that have a size larger than $S_{min}/2$, and n_{MS} denote the number of medium jobs that have a size equal to $S_{min}/2$. Then we can present the heuristic for solving the MILP model in the following steps:

Step 1. Compute LB and the smallest possible number of batches $\underline{i} = \max_{m \in M} \left\{ \left\lceil \sum_j s_j / S_{min} \right\rceil, n_{LS} + \lceil n_{MS}/2 \rceil \right\}$. Set $i = \underline{i}$. Apply MILP with given $\#B = i$ to compute C^i_{max}. If it is infeasible, set $C^i_{max} = \infty$. If $C^i_{max} = LB$, then the resulting makespan is optimal and stop.

Table 2
Solution results for Example 2

Solution procedure	No. of batches	No. of binary variables	$C^i_{\max}$	CPU time (s)
Heuristic	4	40	45	3
	5	50	45	7
	6	60	48	42
Total				52
Improved MILP	≤9	90	45	404
Johnson's algorithm	10	–	79	0

Step 2. Set $i = i + 1$. Apply MILP with given i to obtain $C^i_{\max}$.

Step 3. If $C^i_{\max} > C^{i-1}_{\max}$, stop; the resulting makespan is $C^{i-1}_{\max}$. Otherwise, return to Step 2.

We now elaborate the above procedure. In Step 1, we compute LB by the steps as described above. Then, we determine the smallest possible number of batches $\underline{i}$ by the stated equation. The first part of the maximum is obvious. In the second part, the number of batches is at least $n_{LS} + \lceil n_{MS}/2 \rceil$ because each batch can contain at most one large job and at most two medium jobs. In Example 1, $\underline{i} = \max\{\lceil 36/10 \rceil, \lceil 36/10 \rceil, \lceil 3/2 \rceil\} = 4$. Next, we solve the MILP model with given $\#B = \underline{i}$. The MILP with a given number of batches can be solved much easier because the number of binary variables X_{jb} becomes much smaller. In Example 1, the number of X_{jb} for $\#B = 4$ is only 40, compared to 100 without a given number of batches. However, the MILP model with given $\#B = \underline{i}$ may be infeasible. In this case, we set $C^i_{\max} = \infty$. In Steps 2 and 3, we try to find a local minimum makespan with respect to $\#B$ and stop the algorithm once the minimum is found.

Example 2. Applying the heuristic to the set of jobs in Example 1 yields the result shown in Table 2. The V-shape property does hold here because $C_{\max} = 45, 45, 48, 56, 62, 71, 79$ for $\#B = 4, 5, \ldots, 10$. As a comparison, we also solve the MILP model with $\#B \leq 9$ (and compare with the solution for $\#B = 10$ from Johnson's algorithm) to obtain the optimal solution. It is observed that the heuristic requires only about 13% of the CPU time of the optimal MILP model.

3. Problem with zero intermediate storage

In this section, we consider the problem with zero intermediate storage.

3.1. DS2 model

To formulate the problem as an MILP model, Damodaran and Srihari [2] replace constraints (5)–(11) in DS1 model by the following constraints:

$$C_{11} = P_{11} = D_{11} \tag{15}$$

$$C_{b1} = D_{b-1,1} + P_{b1} \quad \forall b \in B/\{1\} \tag{16}$$

$$C_{b2} = D_{b1} + P_{b2} \quad \forall b \in B \tag{17}$$

$$D_{b1} \geq C_{b-1,2} \quad \forall b \in B/\{1\}. \tag{18}$$

The above constraints have been improved according to Section 2.2, i.e., the variable Q_{km} has been replaced by P_{bm}. Constraints (16) and (17) are used to compute the completion time of batch b on machine 1. Constraint (18) ensures that batch b may leave machine 1 only after batch $b - 1$ has completed its processing on machine 2.

By examining the DS2 model, we note that the following additional constraint is required:

$$D_{b1} \geq C_{b1} \quad \forall b \in B. \tag{19}$$

Otherwise, it may occur that $D_{b1} < C_{b1}$, which is invalid for the problem.

Table 3
Experimental design

Factors	Levels
Job processing times	$p_{jm} \in [1, 100]$
Capacities of machines	$S_m = 10,\ m = 1, 2$
Job sizes	$s_j \in [a_{\min}, a_{\max}]$
Distribution I	$[a_{\min}, a_{\max}] = [1, 5]$
Distribution II	$[a_{\min}, a_{\max}] = [4, 10]$
Distribution III	$[a_{\min}, a_{\max}] = [1, 10]$
Number of instances per combination	10
Number of jobs (first experiment)	$n = 5, 6, 7, 8$
Number of jobs (second experiment)	$n = 10, 15$

3.2. Improved DS2 model

We can use the same approach as in Section 2.2 to improve the DS2 model. Moreover, the model can be further improved by replacing the two variables C_{bm} and D_{bm} with a single variable T_{bm}, which denotes the starting time of batch b on machine m. Accordingly, we replace all the related constraints and objective by the following:

$$\text{Minimize } T_{n+1,2} \tag{20}$$

$$\text{Subject to } T_{21} = P_{11} \tag{21}$$

$$T_{bm} \geq T_{b-1,m} + P_{b-1,m} \quad \forall b = 2, \ldots, n+1,\ m \in M \tag{22}$$

$$T_{b1} = T_{b-1,2} \quad \forall b = 2, \ldots, n+1. \tag{23}$$

Constraints (21) and (22) compute the starting time of batch b on machine m. Constraint (23) ensures that the time that machine 1 starts with a new batch is exactly the same as the time that machine 2 starts with the batch just released from machine 1. Here, we use a dummy variable $T_{n+1,m}$, which not only denotes the objective $C_{\max}$ but also specifies the starting time of the last batch on machine 2. With this improvement, the number of related variables is reduced from $2 \times (\#B)$ to $\#B$, and the number of related constraints is reduced from $4 \times (\#B)$ to $3 \times (\#B)$.

4. Computational experiments

In this section, we verify the performance of the MILP models and the MILP-based heuristic by using the same problem generating scheme as Uzsoy [8]. Job processing times were randomly generated from a discrete uniform distribution $U(1, 100)$. The capacities of both machines were assumed to be 10. Job sizes were generated from discrete uniform distributions between $a_{\min}$ and $a_{\max}$. Three different distributions of job sizes were experimented. In distribution I, we set $(a_{\min}, a_{\max}) = (1, 5)$, which represents the case where job sizes are relatively small so that more jobs can be assigned to a batch. In distribution II, we set $(a_{\min}, a_{\max}) = (4, 10)$, which represents the situation where job sizes are relatively large so that few jobs (only one job in many batches) can be batched together. In distribution III, we set $(a_{\min}, a_{\max}) = (1, 10)$, which represents the case where job sizes are distributed widely. The experimental design including the factors and the levels is summarized in Table 3. The MILP models, generated by a computer program, were solved by LINGO 8.0 and run on a Pentium IV 2.4 GHz (Core 2 Quad) PC.

In the first experiment, we compare the two DS models with their respective improved models. The computational results for the compared models are summarized in Table 4, which provide the information on the CPU times (in seconds) of the models with $n = 5, 6, 7, 8$ where job sizes were generated by distribution I. For each factor combination, 10 independent instances were generated, resulting in a total of 40 instances. The results indicate that the improved models are much better than the original ones, which can only solve a maximum of 7 jobs within 12 h (43 200 s).

In the second experiment, we evaluated the performance of the improved MILP models and the heuristics by solving the problem with $n = 10$ and 15. For each factor combination, 10 independent instances were generated, resulting in a total of 60 instances for each MILP model. The computational results of instances with unlimited and zero intermediate storage are summarized in Tables 5 and 6, respectively. The tables provide the information on the smallest possible number of batches ($\underline{i}$), optimal number of batches (i^*), CPU times for the improved MILP models

Table 4
Comparison of DS models and improved DS models

Instances	n	CPU time (s)		CPU time (s)	
		DS1	Improved DS1	DS2	Improved DS2
1	5	51	0	11	0
2		59	0	9	0
3		68	0	40	0
4		11	0	7	0
5		11	0	8	0
6		43	0	11	0
7		34	0	9	0
8		9	0	6	0
9		68	0	24	0
10		11	0	6	0
11	6	886	0	427	0
12		601	0	296	0
13		111	0	93	0
14		345	0	237	0
15		135	0	113	0
16		272	0	278	0
17		192	0	177	0
18		531	0	178	0
19		509	0	266	0
20		486	0	205	0
21	7	>43 200	1	>43 200	0
22		>43 200	6	>43 200	3
23		>43 200	2	>43 200	1
24		5 291	0	2 513	1
25		14 345	1	10 613	1
26		27 171	1	13 071	0
27		40 712	0	10 713	1
28		29 919	1	10 269	1
29		>43 200	5	>43 200	4
30		8 787	1	7 719	0
31	8	>43 200	7	>43 200	4
32		>43 200	6	>43 200	5
33		>43 200	1	>43 200	2
34		>43 200	2	>43 200	5
35		>43 200	0	>43 200	4
36		>43 200	1	>43 200	1
37		>43 200	4	>43 200	2
38		>43 200	1	>43 200	2
39		>43 200	1	>43 200	2
40		>43 200	3	>43 200	2

with or without the use of lower bound, and CPU times for the heuristic with different specified numbers of batches ($\underline{i}$; $\underline{i}+1$; $\underline{i}+2$). Recall that according to the V-shape property, the heuristic need not compute $C_{\max}^{i+2}$ if $C_{\max}^{i+1} > C_{\max}^{i}$. A review of the results presented in both tables shows that $i^* = \underline{i}$ for most instances, and i^* is very close to $\underline{i}$ ($i^* = \underline{i} + 1$) for the remaining instances. Also, the use of lower bound in the MILP models, in general, can improve the computational efficiency. However, due to the large variation of CPU time, it may increase the CPU time for some instances. From the three columns $\underline{i}$, i^*, and ($\underline{i}$; $\underline{i}+1$; $\underline{i}+2$), we can see that the heuristic can obtain the optimal solution for all the instances for which the optimum can be achieved by MILP. For most instances, the heuristic requires less CPU time than the optimum approach, especially for larger-size problems. However, for some instances especially with distribution II, the heuristic takes longer than the optimum approach. This is because job sizes are relatively large

Table 5
Computational results of instances with unlimited intermediate storage

Inst.	n	Dist.	$\underline{i}$	i^*	Optimum CPU time (s)		Heuristic CPU time (s)		Inst.	n	Dist.	$\underline{i}$	i^*	Optimum CPU time (s)		Heuristic CPU time (s)	
					Without *LB*	With *LB*	$\underline{i}$; $\underline{i}+1$; $\underline{i}+2$	Total						Without *LB*	With *LB*	$\underline{i}$; $\underline{i}+1$; $\underline{i}+2$	Total
1	10	I	4	4	81	167	3; 10	13	31	15	I	4	5	4 479	5 303	12; 47; 1020	1 079
2			3	3	11	11	1;1	2	32			5	5	29 669	32 255	152; 722	874
3			3	3	11	16	0; 1	1	33			5	5	25 636	2 264	85; 233	303
4			3	3	12	11	0; 2	2	34			6	6	21 638	19 471	184; 303	487
5			3	3	7	5	0; 2	2	35			5		>43 200	>43 200	55; 480	535
6			3	3	3	4	0; 2	2	36			5		>43 200	>43 200	22; 631	653
7			4	4	7	10	1; 3	4	37			5		>43 200	>43 200	158; 814; 7302	8 270
8			4	4	43	10	1; 3	4	38			5		>43 200	>43 200	61; 694	755
9			4	4	16	16	1; 2	3	39			5		>43 200	>43 200	18; 339	357
10			3	3	5	8	1; 2	3	40			5	5	>43 200	33 472	98; 375	473
11		II	8	8	549	3	8[c]	8	41		II	13		>43 200	>43 200	>43 200	>43 200
12			9	9	1371	1013	1007; 0[a]	1007	42			13		>43 200	>43 200	>43 200	>43 200
13			7	8	1333	1049	0[b]; 219; 2256	2475	43			12		>43 200	>43 200	>43 200	>43 200
14			8	8	700	1035	155; 852	1007	44			12		>43 200	>43 200	>43 200	>43 200
15			7	8	1298	416	0[b]; 118; 724	842	45			13		>43 200	>43 200	>43 200	>43 200
16			10	10	–	–	0[a]	0	46			13		>43 200	>43 200	>43 200	>43 200
17			8	8	1027	241	140; 498	638	47			12		>43 200	>43 200	0[b]; >43 200	>43 200
18			9	9	440	2	1; 0[a]	1	48			13	13	>43 200	11	21[c]	21
19			9	9	274	1	1; 0[a]	1	49			12		>43 200	>43 200	>43 200	>43 200
20			9	9	1463	3	2; 0[a]	2	50			11		>43 200	>43 200	>43 200	>43 200
21		III	7	7	442	225	13;133	146	51		III	10		>43 200	>43 200	404; >43 200	>43 200
22			5	6	844	373	3; 9; 168	180	52			10		>43 200	>43 200	>43 200	>43 200
23			6	7	249	356	8; 72; 808	888	53			9		>43 200	>43 200	181; 1488	1 669
24			6	7	585	668	0[b]; 122; 395	517	54			8		>43 200	>43 200	40; 332	372
25			5	5	514	116	3; 7	10	55			9		>43 200	>43 200	39032; >43 200	>43 200
26			7	7	170	1	1[c]	1	56			10		>43 200	>43 200	66[b]; >43 200	>43 200
27			8	8	730	1	1[c]	1	57			9		>43 200	>43 200	171; 1491	1 662
28			7	7	280	1	1[c]	1	58			9		>43 200	>43 200	307; 18 104	18 411
29			4	4	93	124	1; 7	8	59			9		>43 200	>43 200	19[b] ; >43 200	8
30			6	6	177	86	10; 62	72	60			7		>43 200	>43 200	355; 1839; 3916	6 110

[a] Solved by Johnson's algorithm.
[b] Infeasible.
[c] The makespan is equal to the lower bound.

in distribution II so that few jobs can be batched together, or equivalently the optimal number of batches i^* is close to n. In such a situation, the heuristic cannot take many advantages of the V-shape property but still has to solve several MILP models, compared to only a single MILP model for the optimum approach. Based on the above discussion, we can safely employ the heuristic as the solution approach for the problem unless the smallest possible number of batches, $\underline{i}$, is very close to n. Finally, we note that problem instances with distribution I is the easiest to solve and instances with distribution II is the hardest. This is consistent with the number of batches in a problem (see $\underline{i}$ and i^* columns).

Table 6
Computational results of instances with zero intermediate storage

Inst.	n	Dist.	$\underline{i}$	i^*	Optimum CPU time (s)		Heuristic CPU time (s)		Inst.	n	Dist.	$\underline{i}$	i^*	Optimum CPU time (s)		Heuristic CPU time (s)	
					Without *LB*	With *LB*	$\underline{i}$; $\underline{i}+1$; $\underline{i}+2$	Total						Without *LB*	With *LB*	$\underline{i}$; $\underline{i}+1$; $\underline{i}+2$	Total
1	10	I	4	4	91	10	2; 6	8	31	15	I	4	4	3 176	1 189	15; 106	121
2			3	3	9	6	1; 1	2	32			5	6	3 485	2 069	124; 148; 1809	2 081
3			3	3	40	6	0; 1	1	33			5	5	17 761	2 155	53; 183	236
4			3	3	10	5	1; 1	2	34			6	6	7 472	6 611	173; 191	364
5			3	3	14	5	0; 1	1	35			5	5	7 239	9 107	47; 247	294
6			3	3	3	5	1; 2	3	36			5	5	6 353	10 483	72; 226	298
7			4	4	9	9	1; 3	4	37			5		>43 200	>43 200	108; 264; 462	834
8			4	4	50	12	1; 5	6	38			5		>43 200	>43 200	21; 532; 6322	6 875
9			4	4	48	15	1; 3	4	39			5		>43 200	>43 200	27; 1171; 6862	7 060
10			3	3	12	4	1; 1	2	40			5	5	>43 200	7 037	30; 192	222
11		II	8	8	234	220	45; 198	243	41		II	13		>43 200	>43 200	>43 200	>43 200
12			9	9	276	202	200; 0[a]	200	42			13		>43 200	>43 200	>43 200	>43 200
13			7	8	688	390	0[b]; 194; 560	754	43			12		>43 200	>43 200	>43 200	>43 200
14			8	8	223	238	162; 272	434	44			12		>43 200	>43 200	>43 200	>43 200
15			7	8	207	261	0[b]; 86; 495	581	45			13		>43 200	>43 200	>43 200	>43 200
16			10	10	–	–	0[a]	0	46			13		>43 200	>43 200	>43 200	>43 200
17			8	8	198	286	47; 348	395	47			12		>43 200	>43 200	0[b]; >43 200	>43 200
18			9	9	237	24	28; 0[a]	28	48			13		>43 200	>43 200	>43 200	>43 200
19			9	9	204	116	80; 0[a]	80	49			12		>43 200	>43 200	>43 200	>43 200
20			9	9	350	397	427; 0[a]	427	50			11		>43 200	>43 200	0[b]; >43 200	>43 200
21		III	7	7	220	174	11; 43	54	51		III	10		>43 200	>43 200	289; 4253; >43 200	>43 200
22			5	6	120	110	0[b]; 12; 96	108	52			10		>43 200	>43 200	>43 200	>43 200
23			6	7	104	135	7; 47;	54	53			9		>43 200	>43 200	150; 11 779	11 929
24			6	7	170	212	0[b]; 18; 152	170	54			8		>43 200	>43 200	9[c]	9
25			5	5	310	21	1; 4	5	55			9		>43 200	>43 200	831; 8504	9 335
26			7	7	56	69	25; 63	88	56			10		>43 200	>43 200	50[b]; >43 200	>43 200
27			8	8	121	130	54; 187	241	57			9		>43 200	>43 200	218; 2580	2 798
28			7	7	140	88	10; 68	78	58			9		>43 200	>43 200	269; 19 859	20 128
29			4	4	54	9	1; 5	6	59			9		>43 200	>43 200	6[b]; 2454; 16 858	19 318
30			6	6	121	34	8; 23	31	60			7	7	32 125	29 269	12; 160	172

[a] Solved by Johnson's algorithm.
[b] Infeasible.
[c] The makespan is equal to the lower bound.

5. Conclusions

In this paper we have improved existing MILP models for scheduling jobs in a flowshop with two batch processing machines such that the makespan is minimized. Computational results have demonstrated that the improved models are much better than the original ones. In specific, many problem instances that cannot be solved within 12 h by the original models can now be resolved by the improved models in less than 7 s. Moreover, we have observed a so-called V-shape property, which is a conjecture since we cannot prove it but, on the other hand, we have not found any counter-example. Based on the V-shape property, an MILP-based heuristic is developed for the problem. Computational experiments have shown that the heuristic can obtain the optimal solutions for all the test problem instances. Future research may be conducted to further improve the MILP models for the batching machine problem. As the heuristic is still an integer program, where the computation time grows exponentially with problem size, it is desired to develop a (pseudo-)polynomial time heuristic in the further research.

References

[1] P. Brucker, A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, S.L. van de Velde, Scheduling a batching machine, Journal of Scheduling 1 (1998) 31–54.
[2] P. Damodaran, K. Srihari, Mixed integer formulation to minimize makespan in a flow shop with batch processing machines, Mathematical and Computer Modelling 40 (2004) 1465–1472.
[3] D. Danneberg, T. Tautenhahn, F. Werner, A comparison of heuristic algorithms for flow shop scheduling problems with setup times and limited batch size, Mathematical and Computer Modelling 29 (1999) 101–126.
[4] M. Mathirajan, Heuristic scheduling algorithms for parallel heterogeneous batch processors, Ph.D. dissertation, Indian Institute of Science, Bangalore, India, 2000.
[5] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, 2nd edition, Prentice-Hall, New Jersey, 2002.
[6] C.N. Potts, M.Y. Kovalyov, Scheduling with batching: A review, European Journal of Operational Research 120 (2000) 228–249.
[7] C.S. Sung, Y.H. Kim, Minimizing due date related performance measures on two batch processing machines, European Journal of Operational Research 147 (2003) 644–656.
[8] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes, International Journal of Production Research 32 (1994) 1615–1635.